



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

Found 277 of

[prediction](#) [global](#) [branch](#) [flush](#) [recovery](#) [recover](#) [recovering](#) [restore](#) [restoring](#) [restores](#)

171,143

Sort results  
by

relevance

Display  
results

expanded form

[Save results to a Binder](#)[Search Tips](#)Open results in a new  
windowTry an [Advanced Search](#)Try this search in [The ACM Guide](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

### 1 [Session 2: value: Fast branch misprediction recovery in out-of-order superscalar](#)

[processors](#)

Peng Zhou, Soner Önder, Steve Carr

 June 2005 **Proceedings of the 19th annual international conference on  
Supercomputing ICS '05**

Publisher: ACM Press

Full text available:  [pdf\(391.00 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Current trends in modern out-of-order processors involve implementing deeper pipelines and a large instruction window to achieve high performance. However, as pipeline depth increases, the branch misprediction penalty becomes a critical factor in overall processor performance. Current approaches to handling branch mispredictions either incrementally roll back to in-order state by waiting until the mispredicted branch reaches the head of the reorder buffer, or utilize checkpointing at branches fo ...

**Keywords:** branch misprediction, checkpoint, processor state, recovery

### 2 [Lightweight recoverable virtual memory](#)



M. Satyanarayanan, Henry H. Mashburn, Puneet Kumar, David C. Steere, James J. Kistler

 December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the  
fourteenth ACM symposium on Operating systems principles SOSP  
'93**, Volume 27 Issue 5

Publisher: ACM Press

Full text available:  [pdf\(1.53 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citing](#), [index terms](#)

*Recoverable virtual memory* refers to regions of a virtual address space on which transactional guarantees are offered. This paper describes *RVM*, an efficient, portable, and easily used implementation of recoverable virtual memory for Unix environments. A unique characteristic of RVM is that it allows independent control over the transactional properties of atomicity, permanence, and serializability. This leads to considerable flexibility in the use of RVM, potentially enlarging the ...

### 3 [Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation](#)


 Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler, David Blaauw, Todd Austin, Krisztian Flautner, Trevor Mudge  
December 2003

## Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture

**Publisher:** IEEE Computer Society

Full text available:  pdf(568.17 KB) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

With increasing clock frequencies and silicon integration, power aware computing has become a critical concern in the design of embedded processors and systems-on-chip. One of the more effective and widely used methods for power-aware computing is dynamic voltage scaling (DVS). In order to obtain the maximum power savings from DVS, it is essential to scale the supply voltage as low as possible while ensuring correct operation of the processor. The critical voltage is chosen such that under a worst-case ...

### 4 Processor microarchitecture II: Predicate prediction for efficient out-of-order execution



Wei-haw Chuang, Brad Calder

June 2003 **Proceedings of the 17th annual international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  pdf(220.52 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Predicated execution is an important optimization even for an out-of-order processor, since it can eliminate hard to predict branches and help to enable software pipelining. Using predication with out-of-order execution creates a naming bottleneck, because there can be multiple definitions reaching a use, and not knowing which use is the correct one can stall the processor. In this paper, we examine using predicate prediction to speculatively allow execution to proceed in the face of multiple def ...

**Keywords:** predicate prediction, predicated execution

### 5 Merging path and gshare indexing in perceptron branch prediction



David Tarjan, Kevin Skadron

September 2005 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 2 Issue 3

**Publisher:** ACM Press

Full text available:  pdf(465.68 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We introduce the *hashed* perceptron predictor, which merges the concepts behind the gshare, path-based and perceptron branch predictors. This predictor can achieve superior accuracy to a path-based and a global perceptron predictor, previously the most accurate dynamic branch predictors known in the literature. We also show how such a predictor can be ahead pipelined to yield one cycle effective latency. On the SPECint2000 set of benchmarks, the hashed perceptron predictor improves accuracy ...

**Keywords:** Branch prediction, neural networks, two-level predictors

### 6 GPGPU: general purpose computation on graphics hardware



David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes GRAPH '04**





**Publisher:** ACM Press

Full text available:  pdf(63.03 MB) Additional Information: [full citation](#), [abstract](#)



The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable

vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

## 7 A study of slipstream processors

 Zach Purser, Karthik Sundaramoorthy, Eric Rotenberg  
December 2000 **Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture**  
**Publisher:** ACM Press  
Full text available:  [pdf\(130.26 KB\)](#)  
 [ps\(398.01 KB\)](#) Additional Information: [full citation](#), [references](#), [citing](#)s, [index terms](#)  
 [Publisher Site](#)



## 8 Processor microarchitecture I: Recycling waste: exploiting wrong-path execution to improve branch prediction

 Haitham Akkary, Srikanth T. Srinivasan, Konrad Lai  
June 2003 **Proceedings of the 17th annual international conference on Supercomputing**  
**Publisher:** ACM Press  
Full text available:  [pdf\(311.39 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Despite continuous improvement in branch prediction algorithms, branch misprediction remains a major limitation on microprocessor performance. As pipelines are widened or stretched deeper, branch prediction will become even more crucial. This paper taps into a currently wasted resource, wrong-path execution, to help improve branch prediction. Due to control independence, often the outcomes of branches that are executed along the wrong-path match the outcomes on the correct-path. Current branch p ...

**Keywords:** branch prediction, deep pipelines, instruction reuse

## 9 Hardware fault containment in scalable shared-memory multiprocessors

 Dan Teodosiu, Joel Baxter, Kinshuk Govil, John Chapin, Mendel Rosenblum, Mark Horowitz  
May 1997 **ACM SIGARCH Computer Architecture News , Proceedings of the 24th annual international symposium on Computer architecture ISCA '97**, Volume 25 Issue 2  
**Publisher:** ACM Press  
Full text available:  [pdf\(2.05 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citing](#)s, [index terms](#)

Current shared-memory multiprocessors are inherently vulnerable to faults: any significant hardware or system software fault causes the entire system to fail. Unless provisions are made to limit the impact of faults, users will perceive a decrease in reliability when they entrust their applications to larger machines. This paper shows that fault containment techniques can be effectively applied to scalable shared-memory multiprocessors to reduce the reliability problems created by increased mach ...

## 10 Dynamic dead-instruction detection and elimination

 J. Adam Butts, Guri Sohi  
October 2002 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , Proceedings of the 10th international conference on Architectural support for programming languages and operating systems ASPLOS-X**, Volume 36 , 37 , 30 Issue 5 , 10 , 5  
**Publisher:** ACM Press

Full text available:  pdf(1.50 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

We observe a non-negligible fraction--3 to 16% in our benchmarks--of *dynamically dead instructions*, dynamic instruction instances that generate unused results. The majority of these instructions arise from static instructions that also produce useful results. We find that compiler optimization (specifically instruction scheduling) creates a significant portion of these *partially dead* static instructions. We show that most of the dynamically instructions arise from a small set of st ...


# 11 A comparison of three current superscalar designs



Michael Laird

June 1992 **ACM SIGARCH Computer Architecture News**, Volume 20 Issue 3

**Publisher:** ACM Press

Full text available:  pdf(824.41 KB) Additional Information: [full citation](#), [abstract](#), [index terms](#)

A standardized view of superscalar architectures is presented, and three current superscalar designs are compared using this framework. The designs studied are the Metaflow Light-ning SPARC, the IBM RS/6000, and the Intel i960MM.

# 12 Status report of the graphic standards planning committee



Computer Graphics staff

August 1979 **ACM SIGGRAPH Computer Graphics**, Volume 13 Issue 3

**Publisher:** ACM Press

Full text available:  pdf(15.01 MB) Additional Information: [full citation](#), [references](#), [citations](#)

# 13 Prophet/Critic Hybrid Branch Prediction



Ayose Falcon, Jared Stark, Alex Ramirez, Konrad Lai, Mateo Valero

March 2004 **ACM SIGARCH Computer Architecture News , Proceedings of the 31st annual international symposium on Computer architecture ISCA '04**,  
Volume 32 Issue 2

**Publisher:** IEEE Computer Society, ACM Press

Full text available:  pdf(214.72 KB) Additional Information: [full citation](#), [abstract](#), [citations](#)

This paper introduces the prophet/critic hybrid conditional branch predictor, which has two component predictors that play the role of either prophet or critic. The prophet is a conventional predictor that uses branch history to predict the direction of the current branch. Further accesses of the prophet yield predictions for the branches following the current one. Predictions for the current branch and the ones that follow are collectively known as the branch's future. They are actually a prophecy, or pred ...

# 14 Architecture 2: Dual path instruction processing



Juan L. Aragón, José González, Antonio González, James E. Smith

June 2002 **Proceedings of the 16th international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  pdf(332.19 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The reasons for performance losses due to conditional branch mispredictions are first studied. Branch misprediction penalties are broken into three categories: pipeline-fill penalty, window-fill penalty, and serialization penalty. The first and third of these produce most of the performance loss, but the second is also significant. Previously proposed dual (or multi) path execution methods attempt to reduce all three penalties, but these methods are also quite complex. Most of the complexity is ...



**Keywords:** branch misprediction penalty, confidence estimation, dual path processing, pre-scheduling

15 The use of multithreading for exception handling

Craig B. Zilles, Joel S. Emer, Gurindar S. Sohi

November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**

**Publisher:** IEEE Computer Society

Full text available:  pdf(1.49 MB)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)  
[Publisher Site](#)

Common hardware exceptions, when implemented by trapping, unnecessarily serialize program execution in dynamically scheduled superscalar processors. To avoid the consequences of trapping the main program thread, multithreaded CPUs can exploit control and data independence by executing the exception handler in a separate hardware context. The main thread doesn't squash instructions after the excepting instruction, conserving fetch bandwidth and allowing execution of instructions inde ...

16 Full-system timing-first simulation



Carl J. Mauer, Mark D. Hill, David A. Wood

June 2002 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '02**, Volume 30 Issue 1

**Publisher:** ACM Press

Full text available:  pdf(87.83 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

Computer system designers often evaluate future design alternatives with detailed simulators that strive for *functional fidelity* (to execute relevant workloads) and *performance fidelity* (to rank design alternatives). Trends toward multi-threaded architectures, more complex micro-architectures, and richer workloads, make authoring detailed simulators increasingly difficult. To manage simulator complexity, this paper advocates decoupled simulator organizations that separate functiona ...


17 The evolution of Coda



M. Satyanarayanan

May 2002 **ACM Transactions on Computer Systems (TOCS)**, Volume 20 Issue 2

**Publisher:** ACM Press

Full text available:  pdf(441.35 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Failure-resilient, scalable, and secure read-write access to shared information by mobile and static users over wireless and wired networks is a fundamental computing challenge. In this article, we describe how the Coda file system has evolved to meet this challenge through the development of mechanisms for server replication, disconnected operation, adaptive use of weak connectivity, isolation-only transactions, translucent caching, and opportunistic exploitation of hardware surrogates. For eac ...

**Keywords:** Adaptation, Linux, UNIX, Windows, caching, conflict resolution, continuous data access, data staging, disaster recovery, disconnected operation, failure, high availability, hoarding, intermittent networks, isolation-only transactions, low-bandwidth networks, mobile computing, optimistic replica control, server replication, translucent cache management, weakly connected operation

18



Novel ideas: Performance characterization of a hardware mechanism for dynamic optimization

Brian Fahs, Satarupa Bose, Matthew Crum, Brian Slechta, Francesco Spadini, Tony Tung,

Sanjay J. Patel, Steven S. Lumetta

December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**

**Publisher:** IEEE Computer Society

Full text available:  [pdf\(1.31 MB\)](#)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)  
[Publisher Site](#)

We evaluate the rePLay microarchitecture as a means for reducing application execution time by facilitating dynamic optimization. The framework contains a programmable optimization engine coupled with a hardware-based recovery mechanism. The optimization engine enables the dynamic optimizer to run concurrently with program execution. The recovery mechanism enables the optimizer to make speculative optimizations without requiring recovery code. We demonstrate that a rePLay configuration performing ...



## 19 Cherry-MP: Correctly Integrating Checkpointed Early Resource Recycling in Chip Multiprocessors



Meyrem Kyrman, Nevin Kyrman, Jose F. Martinez

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

**Publisher:** IEEE Computer Society

Full text available:  [pdf\(453.38 KB\)](#)  Additional Information: [full citation](#), [abstract](#)  
[Publisher Site](#)

Checkpointed Early Resource Recycling (Cherry) is a recently-proposed micro-architectural technique that aims at improving critical resource utilization by performing aggressive resource recycling decoupled from instruction retirement, using a checkpoint/rollback mechanism to recover from occasional incorrect execution. In this paper, we explore correctness and performance issues that arise when Cherry-enabled processors are used in chip multiprocessor architectures. We propose mechanisms to address ...



## 20 How to Fake 1000 Registers



David W. Oehmke, Nathan L. Binkert, Trevor Mudge, Steven K. Reinhardt

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

**Publisher:** IEEE Computer Society

Full text available:  [pdf\(318.71 KB\)](#)  Additional Information: [full citation](#), [abstract](#)  
[Publisher Site](#)

Large numbers of logical registers can improve performance by allowing fast access to multiple subroutine contexts (register windows) and multiple thread contexts (multithreading). Support for both of these together requires a multiplicative number of registers that quickly becomes prohibitive. We overcome this limitation with the virtual context architecture (VCA), a new register-file architecture that virtualizes logical register contexts. VCA works by treating the physical registers as a cache ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)



Welcome United States Patent and Trademark Office

Search Session History

[BROWSE](#)

[SEARCH](#)

[IEEE XPLORE GUIDE](#)

Thu, 16 Mar 2006, 11:23:57 AM EST

Edit an existing query or compose a new query in the Search Query Display.

Search Query Display

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

[Run Search](#) [Reset](#)

Recent Search Queries

- #1

(((global <and> (recover\* <or> restor\* <or> correcting <or> corrected) <and> branch))<in>metadata)
- #2

(((global <and> (recover\* <or> restor\* <or> correcting <or> corrected) <and> branch <and> predict\*))<in>metadata)
- #3

(((global <and> (recover\* <or> restor\* <or> correcting <or> corrected) <and> branch))<in>metadata)
- #4

(((global <and> (recover\* <or> restor\* <or> correcting <or> corrected) <and> branch <and> predict\*))<in>metadata)
- #5

(((global <and> (updat\*) <and> branch <and> predict\*))<in>metadata)
- #6

(((global <and> (updat\*) <and> branch <and> predict\*))<in>metadata)
- #7

(((micro <and> rollback <and> flush\* <and> predict\*))<in>metadata)
- #8

(((micro <and> rollback <and> tremblay))<in>metadata)

[Clear Session History](#)

